8-2003

UbiTour: a 3G/WLAN Architecture to Support E-Tourism

Vivek Chinta
University of New Orleans, vivek.chinta@gmail.com

UBITOUR : A 3G/WLAN ARCHITECTURE TO SUPPORT E-TOURISM

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
The Department of Computer Science

by

Vivek Chinta

B.S., Chaitanya Bharathi Institute of Technology, 1996
M.S., University of New Orleans, 2000

August 2003

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Golden G. Richard III for supervising my thesis. His intelligent ideas, eagerness to help and provide good suggestions and his invaluable support has helped me complete my thesis successfully.

I would also like to thank Dr. Ming – Hsing Chiu and Dr. Adlai DePano for their guidance and support.

I extend my thanks to Dr. Mahdi Abdelguerfi for his support and keen interest shown during the course of my thesis.

I would like to thank every faculty and staff member in the Computer Science. Completion of my work was made possible by the excellent laboratories maintained by the Department where I could work in a good atmosphere.

Finally, I would like to thank Niharika and Ravi for their mental and intellectual support. Their valuable suggestions and ideas played a major role in the successful completion of my thesis.

TABLE OF CONTENTS

ABSTRACT

Electronic tourism is a new type of application that provides information, often location-based, to tourists. The information includes route guidance, listings of nearby points of interest, guided tours with multimedia presentations and experiences shared by visitors. E-tourism applications can also include storing sound and images recorded by visitors. An architecture to support e-tourism called 'UbiTour' is presented in this dissertation. UbiTour provides different services to visitors such as location-based information, route-guidance, announcements and visitor collaboration.

UbiTour aims at using 3G for ubiquitous connectivity and WLAN for high-speed, local information, running on a PDA. WLAN, specifically 802.11, provides a high-bandwidth, broadcast-based connection over a very limited area, while 3G provides a ubiquitous, though lower speed, connection across large metropolitan areas. The services provided by UbiTour have different requirements and constraints, some better suited for WLAN and others for 3G. By seamlessly combining the two technologies, UbiTour can provide a powerful electronic tour guide.

CHAPTER 1

INTRODUCTION

The widespread deployment of two complementary technologies, 3G cellular networks and 802.11-based wireless local area networks (WLANs) [4], will enable a number of mobile applications that are difficult or impractical to deploy without ubiquitous connectivity. 802.11-based networks provide high bandwidth, low latency connectivity on a small scale; because of deployment costs, 802.11-based WLANs will not be deployed over entire metropolitan areas. 3G cellular networks provide moderate latency connectivity on a metropolitan, national or global scale, but currently offer bandwidth that is several orders of magnitude lower than 802.11.

In this report, an architecture called UbiTour is described for supporting electronic tourism. UbiTour provides various services to visitors such as location based information, route guidance, and visitor collaboration such as sharing experiences. UbiTour aims to integrate 802.11 and 3G cellular connectivity, emphasizing the strengths of each wireless technology. 802.11 networks can be deployed on a limited scale to support high bandwidth operations such as delivery of video and broadcast-based operations such as electronic advertising, while 3G connectivity can be used to support always-connected, low bandwidth operations. It is believed that this type of integration, using 3G to bridge gaps in WLAN connectivity, will become a popular model for ubiquitous computing applications.

In the remainder of this chapter, electronic tourism is discussed in more detail, 802.11 and 3G cellular networks are looked at more deeply and finally some related work in electronic tourism is discussed. Chapter 2 discusses the issues involved in implementing a sound foundation over which the framework of UbiTour is built. Chapter 3 provides a brief overview of the services provided by UbiTour. Chapter 4 presents the architecture of UbiTour. The design and implementation issues are discussed in Chapter 5.

1.1 Electronic Tourism

Electronic tourism aims to allow more freestyle exploration of a tourist attraction or city than traditional "canned" tours, while providing tools for enhancing the tourist's experience. A location-aware electronic tour guide is an application that runs on a mobile device such as a PDA or small laptop, which a tourist carries while visiting a tourist area. At a minimum, the tour guide should display information about attractions to the tourist based on his current location within the tourist area. It must also support some navigational services like providing directions to go from one place to another. The ability to display maps of the areas surrounding the tourist's current location is an added advantage. Other desirable features include collaborative functions for tourist groups, electronic advertisements that provide up-to-date menus for local restaurants and shopping information, and support for capturing digital images and video for later viewing. Location-based services are guided by the tourist's current location, obtained using a global positioning system (GPS) device. The GPS device is attached to the electronic tour guide.

Since mobile devices are resource constrained, the bulk of the processing must be done remotely on dedicated servers and client devices must be updated periodically. In

addition, group functions, which allow tourists visiting a city together to collaborate during their exploration, require ubiquitous connectivity. These constraints necessitate the existence of a wireless network that spans an entire tourist area, making 3G an obvious choice. The information base, containing multimedia content, maps, and the like, can then be stored on remote servers, and information will be downloaded to the client devices on demand. While 3G connectivity can support many important features in an architecture for electronic tourism, higher bandwidth WLANs also have a place. For streaming video and audio related to specific tourist sites, WLANs are much more appropriate. WLANs are also preferable for uploading digital images and videos captured by visitors for later viewing. In addition, electronic advertisements and other location-dependent content are better distributed using broadcast (to be precise, *geocasting*, which is broadcast constrained to a specific geographical area). By using a location-limited broadcast, the advertiser can more accurately target the information they present to potential customers and avoid wasting bandwidth (and their money) annoying others. Current generation 3G networks provide no broadcast capabilities, but 802.11 is inherently a broadcast-based medium. A local WLAN associated with each major attraction can broadcast advertising, announcements, and multimedia content to nearby electronic tour guides. Seamless roaming between the 3G and WLAN portions of the network provides constant connectivity for the tourist application. The integration of WLAN and 3G in an architecture for supporting electronic tourism allows more flexibility in the types of services provided to the tourist than systems based on only cellular connectivity (where bandwidth is still somewhat scarce, but uninterrupted service can be provided) or only on WLANs (where ubiquitous connectivity is usually impossible).

The UbiTour architecture aims to utilize the best features of both 802.11-based WLANs and 3G cellular networks, providing a tourist with ubiquitous connectivity and access to services. The prototype implementation of the architecture targets the Math Building and its surrounding buildings in the University of New Orleans Campus. It provides context-sensitive information about points of interest and directions between the points of interest. UbiTour also supports collaboration among groups of visitors, allowing them to see locations of other visitors and share user experiences. The architecture supports broadcast of announcements over 802.11 WLANs, aimed at providing both informational messages and paid advertisements (in a commercial deployment). The UbiTour system serves as an ideal testbed for investigating the integration of services over 3G and WLAN networks, since a vast number of electronic tourism services with very different connectivity, bandwidth, and latency requirements can be imagined.

The architecture is targeted at higher-end mobile computers such as the HP iPaq or Sony PEG-NZ90, equipped with network adaptors for both 3G and WLAN.

1.2 Contrasting 3G with 802.11

The 802.11 family of protocols (802.11a, 802.11b, and 802.11g being the current variations) are wireless LAN protocols providing high bandwidth (11-54Mb/sec) and relatively low deployment cost (on a small scale), while being substantially easier to install than traditional wired networks. This has made 802.11 quite popular and WLANs based on 802.11 are widely deployed in corporate offices, coffee shops, college campuses, and even homes. In addition, WLAN "hotspot" companies like T-Mobile and Boingo Wireless have begun to provide 802.11 connectivity for airports, hotels, and other public places. The downside of 802.11-based WLANs is that they provide a limited

range of geographical coverage, on the order of 500-1000 feet in open spaces, but much lower in closed spaces. Another major drawback associated with WLANs is the associated WEP security protocol, which fails to provide adequate security due to inherent security flaws. Developing standards such as 802.11i promise to improve security but the limited transmission range of 802.11 will remain unchanged.

3G cellular technology overcomes the bandwidth limitations of 2/2.5G cellular networks to provide data rates of up to 2 Mb/sec, making it possible to deliver content-rich services and applications to mobile users. One very significant edge that 3G has over WLANs is that it provides ubiquitous coverage, typically over entire metropolitan areas. Major cellular companies like Monet Mobile, Metro PCS, Verizon Wireless and Sprint PCS have begun to roll out commercial 3G services in stages, with moderate bandwidth available now and substantially higher bandwidth on the way as the networks are upgraded. Several providers, such as Sprint PCS and Verizon, already have flat-rate plans for 3G data service. In fact, locations that would want electronic tourism will generally have a strong 3G infrastructure in place. The measurements on Sprint's 3G network, the one with which we are most familiar, consistently showed data rates of approximately 30-90Kb/sec.

1.3 Current Efforts at Integrating WLAN and 3G

With WLAN already being a popular technology and 3G becoming widespread soon, establishing a common platform where WLAN and 3G could interoperate will provide the combined benefits of high bandwidth and ubiquitous coverage. The existence of such a platform with a seamless roaming solution between 3G and WLAN will enable users to move freely from one network to the other without the hassles of manually modifying connection settings and reconnecting. The provisioning of persistent

connections makes it possible to deliver uninterrupted services to the users as they roam across networks. Such a platform would require intelligent sensing mechanisms in the client side devices that can dynamically switch to WLANs whenever and wherever available for high bandwidth and at other times remain on 3G for continued connectivity.

Considerable research is already being carried out in this direction and a handful of companies like Lucent, T-Mobile, ipUnplugged and Airvana have successfully demonstrated seamless handoffs between 3G and WLAN networks. Nokia has developed a dual mode PC Card, D311, that integrates both GSM/GPRS and 802.11b WLAN technologies. The PC Card achieves data rates of up to 40 kbps in GPRS networks and 11 Mbps in WLAN networks.

These endeavors to deploy WLAN 3G roaming are an indication that the "always-and-best-connected" concept, where mobile users will always be connected to the fastest network available, will no longer be a notion but instead a reality in the near future. Complementary use of WLAN and 3G is a very significant component in the UbiTour architecture for supporting location aware, electronic tourism.

1.4 Related Work in Electronic Tourism

A handful of electronic tour guides have been developed and deployed in the past. In this section a brief survey of some of these systems is presented.

The GUIDE system [1], developed at Lancaster University, is one of the most elegant systems. GUIDE is a web-based context-sensitive electronic tour guide that has been deployed in Lancaster, UK. Fujitsu TeamPads are used as the end-user systems, each running a customized web browser for information display. The entire system is based on a cell-based wireless communications infrastructure (compliant with the 802.11 standards), which consists of a set of interconnected cells. Each cell covers a certain area

and has a cell-server that disseminates information to all the end-user systems within the cell, using a specialized broadcast protocol. This broadcast protocol is used in lieu of a common, reliable transport protocol like TCP for reasons of scalability. GUIDE does not use GPS devices; all location information is provided by the cell-servers. Portions of the information model are cached locally on the end-user systems, enabling continued though degraded operation during periods of network disconnection. Some of the salient features of the system include access to context-sensitive information, pre-created tours, interactive services and creation of custom tours. The GUIDE system was later extended to support co-operation among visitors in the form of text-messaging, location information-sharing and viewing comments about the attractions from other visitors. This was made possible by extending the communications infrastructure with Bluetooth [6] micro-cells for more fine-grained position resolution.

The Cyberguide system [2], developed by the Future Computing Environments (FCE) Group at the Georgia Institute of Technology, is a mobile, context-aware tour guide system. Prototypes were built for indoor and outdoor uses. An initial indoor Cyberguide was built on Apple MessagePads, enabling visitors to the GVU (Graphics, Visualization and Usability) open houses to view maps, read information about the demonstrations, and fill out electronic questionnaires, but it lacked any user collaboration features. Position information was obtained using IR beacons. The outdoor version used GPS receivers for positioning. Cyberguide II, a later indoor version, facilitates collaboration by allowing users to see the locations of other users and share comments. CyBARguide, another outdoor version, guides tourists in locating refreshment locations and provides maps, information and navigation facilities.

The Voyager MIT Campus Guide [3], developed at the MIT Media Laboratory, is an electronic guide for the MIT Campus, providing information to visitors about various locations of interest. The end-user system is a HP iPaq H3650 running Windows CE 3.0 with a connected GPS receiver. Because a 2G cellular network was used, the reliance on the network was minimum due to the low bandwidth offered by 2G cellular networks.

CHAPTER 2

LAYING THE FOUNDATION

The basic functionality of an electronic tour guide is to provide location based information. The provision of such location dependent information entails the tour guide with the burden of keeping track of the current location of the visitor who wields the handheld device. The location detection can be achieved using the Global Positioning System (GPS) wherein the handheld device can be interfaced with a GPS receiver which continually receives location updates from the GPS satellites. These location updates can be sent over the network to a remote service which keeps track of the visitors' whereabouts. UbiTour uses this technique to track visitor locations. The remainder of this chapter describes in more detail how this is achieved.

2.1 Mapping the tourist Area

The first step taken in the development of UbiTour was to record the latitude/longitude readings for all the buildings, roads and intersections in the interested area. A set of 4 end-points that encloses all the locations of interest were selected and their readings were taken as well. Let the area enclosed by the 4 end-points be called as the 'Physical Space'. Next, a conversion formula is required to map all the latitude-longitude values to x-y values in a co-ordinate x-y plane. Let the mapped Physical Space be called as the 'Virtual Space'. The mapping from the Physical Space to the Virtual Space is achieved as follows.

The smallest rectangle that encloses the 4 end-points of the Physical Space is first determined. The bottom left corner of the rectangle is fixed as the origin of the x-y plane in the Virtual Space. The latitude value of this origin is fixed as the Reference latitude (LT_{REF}) and its longitude value is fixed as the Reference longitude (LN_{REF}). A point P in the Physical Space with latitude-longitude values (LT_P, LN_P) can then be mapped to the point V in the Virtual Space with values (X_V, Y_V) as –

$$X_V = \textbf{distance between } LN_{REF} \textbf{ and } LN_P$$

$$Y_V = \textbf{distance between } LT_{REF} \textbf{ and } LT_P$$

The calculation of the vertical distance (Y_V) is pretty straightforward because the lines of latitude are equally spaced. The calculation of the horizontal distance (X_V) however needs some tweaking because the lines of longitude are farthest apart at the equator and meet at the poles. This can create errors in the mapping. To keep these errors to a minimum, the latitude value of the center of the enclosing rectangle is determined and the horizontal distance is measured along this central latitude (LT_{CEN}). The mathematical formulae for the X and Y mapping are as follows :

$$X_V = \text{EARTH\_RADIUS} * (LN_P - LN_{REF}) * \cos(LT_{CEN})$$

$$Y_V = \text{EARTH\_RADIUS} * (LT_P - LT_{REF})$$

The latitude and longitude values are expressed in radians. The units of X and Y distances are determined by the units of the radius of the earth. The above formulae are based on a flat- earth approximation. Figure 2.1 shows the mapping for the Math Building from the Physical Space to the Virtual Space :

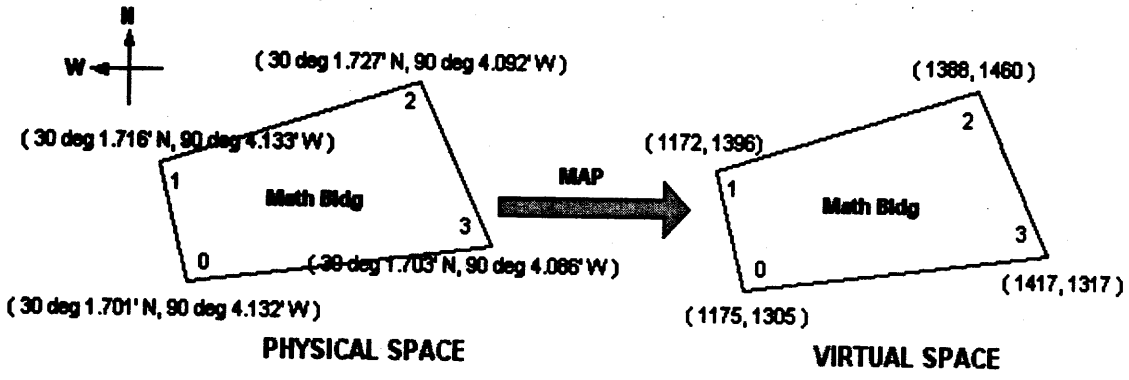

Figure 2.1: Mapping from Physical Space to Virtual Space

2.2 The P2V Mapper Application

To eliminate all manual calculations, the P2V Mapper Application has been developed which maps a given physical space to a virtual space. The application formulates its own mapping functions based on the input data. UbiTour utilizes this application to map the campus area surrounding and including the Math building.

The input to the application is an XML file that contains the latitude-longitude values for all the locations of interest including the values for the 4 end-points of the physical space which enclose all the locations of interest. Parts of the physical space XML file used for the UbiTour system are as shown in Figure 2.2:

```
<PHYSICALSPACE>
.....
.....
<LOCATION>
    <ID> 101 </ID>
    <NAME> Math Building </NAME>
    <READINGS>
        <READING>
            <LATITUDE>
                <DEGREES> 30 </DEGREES>
                <MINUTES> 1.701 </MINUTES>
                <DIRECTION> NORTH </DIRECTION>
            </LATITUDE>
            <LONGITUDE>
                <DEGREES> 90 </DEGREES>
                <MINUTES> 4.132 </MINUTES>
                <DIRECTION> WEST </DIRECTION>
            </LONGITUDE>
        </READING>
        <READING>
            .....
            .....
        </READING>
        .....
        .....
    </READINGS>
</LOCATION>
<LOCATION>
    <ID> 102 </ID>
    <NAME> Liberal Arts Building </NAME>
    <READINGS>
        .....
        .....
    </READINGS>
</LOCATION>
.....
</PHYSICALSPACE>
```

Figure 2.2: The XML Physical Space File

The application generates an XML virtual space file. Parts of the virtual space XML file generated for UbiTour are as shown in Figure 2.3:

```
<VIRTUALSPACE>
.....
<LOCATION>
    <ID>101</ID>
    <NAME>Math Building</NAME>
    <CORNERS>
        <CORNER>
            <X>1175</X>
            <Y>1305</Y>
        </CORNER>

        <CORNER>
            <X>1172</X>
            <Y>1396</Y>
        </CORNER>
        .....
    </CORNERS>
</LOCATION>
.....
</VIRTUALSPACE>
```

Figure 2.3: The generated XML Virtual Space File

2.3 Incorporating GPS Functionality

UbiTour uses GPS for tracking the locations of visitors. This section describes the incorporation of GPS functionality in UbiTour. Section 2.3.1 discusses the interfacing of GPS receivers to the UbiTour handheld. Section 2.3.2 discusses the parsing of GPS information.

2.3.1 Interfacing GPS Receiver

The UbiTour handheld is equipped with a GPS receiver that continually receives location updates from GPS satellites. These location updates are retrieved by the handheld over the serial communications port. A set of configuration parameters are associated with the RS232 interface. These parameters must be set to achieve proper and meaningful communication between the handheld and the receiver. Different GPS receivers have different values for the parameters. To make the client application running on the handheld totally independent of the kind of GPS receiver used, a small utility

application called 'GPSReceiverConfig' is installed on the handheld. Before a specific GPS receiver is connected to the handheld, the utility application must be run to enter the configuration parameter values designated for that GPS receiver. The parameters are stored in a local file, which are later retrieved by the client application. This facilitates different GPS receivers to be connected without any modifications to the client – side code.

A screenshot of the GPSReceiverConfig application is shown in Figure 2.4 which illustrates the serial configuration settings for a Garmin Etrex Legend GPS receiver.

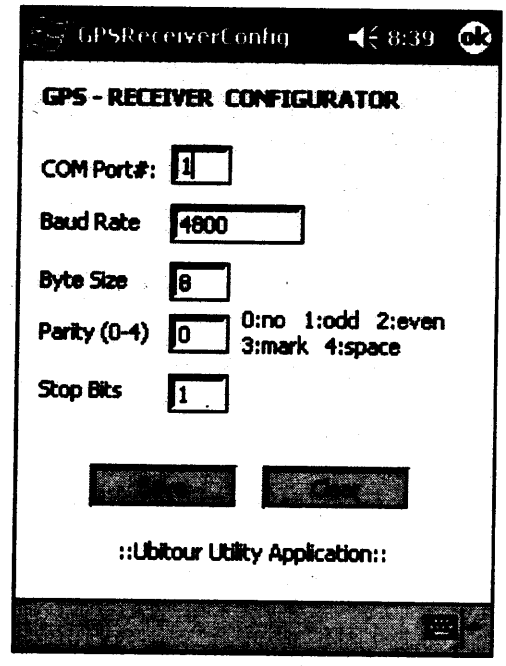

Figure 2.4: GPSReceiverConfig Application

As seen from the screenshot, the 5 serial configuration parameters associated with any GPS receiver are –

1. **COM Port** : The port number over which communication occurs.
2. **Baud Rate** : The rate at which the receiver operates.

3. **Byte Size** : Number of bits in the bytes transmitted and received.
4. **Parity** : The parity scheme used.
5. **Stop Bits** : The number of stop bits to be used.

The values associated with these parameters are different for different GPS receivers. I have successfully been able to use a Garmin Etrex Legend GPS receiver and a Navman I Series GPS receiver with the UbiTour application.

2.3.2 GPS Parsing

Most GPS receivers output location and other data over the serial connection in a standard format known as the NMEA (National Marine Electronics Association) format [7]. In this format, a receiver spits out a set of NMEA sentences periodically. A sentence is a line of data that begins with a '$' and ends with a carriage return/line feed sequence. The '$' is followed by a two letter sequence that defines the device type ('GP' for GPS receivers). This is followed by a three letter sequence that indicates the sentence type. The rest of the information depends on the sentence type. A set of sentence types are defined by the NMEA standard where each type provides some specific information. One specific sentence type that contains the latitude and longitude information is the RMC (Recommended Minimum Configuration). Every GPS receiver that talks NMEA supports the RMC sentence type. So this sentence can be used to extract location information.

A sample RMC sentence received from a GPS receiver is as shown. The pieces of information are separated by a comma, and an explanation of the parts is given.

$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W,A*6A<CR>
<LF>

1. Sentence Type Identifier (Protocol Header) : RMC

2. Time of Fix (UTC) : 123519

3. Status (Validity) : A (A – Active, V – Void)

4. Latitude : 4807.038,N (ddmm.mmm)

5. Longitude : 01131.000,E (dddmm.mmm)

6. Ground Speed (in Knots) : 022.4

7. Track Angle (in Degrees) : 084.4

8. Date : 230394

9. Magnetic Variation : 003.1,W

10. Checksum Data (always begins with *) : *6A

11. Message Termination (carriage return line feed) : <CR><LF>

The sentence needs to be parsed to extract the latitude and longitude fields. A C++ GPS parsing library has been implemented on Windows CE which parses a few important NMEA sentences including the RMC sentence type. The library supports NMEA 0183 version 2.2/2.3 sentences. To parse a sentence, an object of the class NMEAParser needs to be created and the sentence passed to it using the Parse() method. The method returns the type of sentence passed. Based on the returned sentence type, appropriate methods can be invoked to retrieve the various parts or fields of the sentence. The code snippet in Figure 2.5 illustrates the use of the parser library to extract latitude/longitude values from an RMC sentence:

```
//Create a NMEAParser object
NMEAParser Parser ;
//Invoke the Parse method and pass the sentence(of type char*) to be parsed
//The method returns the sentence type
char* sentenceType = Parser.Parse ( sentence ) ;
//If the sentence type is RMC
if ( strcmp ( sentenceType, "RMC") == 0 )
{
        //Invoke methods to extract latitude information
        int latDegrees = Parser.nmeaLatDgs ( ) ;
        double latMinutes = Parser.nmeaLatMns ( ) ;
        char* latDirection = Parser.nmeaLatDir ( ) ;

        //Invoke methods to extract longitude information
        int lonDegrees = Parser.nmeaLonDgs ( ) ;
        double lonMinutes = Parser.nmeaLonMns ( ) ;
        char* lonDirection = Parser.nmeaLonDir ( ) ;
}
```

Figure 2.5: Code snippet illustrating GPS parsing

UbiTour utilizes this library to parse the incoming RMC sentences form the attached GPS receiver on the handheld. The extracted latitude and longitude values are sent over the network to a remote server to be mapped into the Virtual Space.

2.4 Proximity Determination

UbiTour needs to periodically calculate and maintain the proximity of a visitor to the various locations of interest in order to provide the location-sensitive services. The Virtual Space created by the P2V Mapper application has the co-ordinates for all the locations of interest. The co-ordinates of the visitor are now required to determine his proximity to the locations of interest. These co-ordinates are obtained from the physical position of the visitor, which is expressed as a latitude/longitude pair, by applying the mapping formulae described in section 2.2.

To aid the proximity determination process, a utility library called ProxUtils has been implemented that provides certain useful classes. The most useful among them is the Pgon class that can be used to create polygons with given sides and vertices. Every

building in the Virtual Space can be represented as a polygon. So, a Pgon object can be created for every building. The code snippet in Figure 2.6 shows the creation of a polygon for the Math Building. The co-ordinates are hard-coded for readability.

```
//The number of sides
int nSides = 4 ;
//The x co-ordinates of the vertices
int X[ ] = { 1172, 1388, 1417, 1175 } ;
//The y co-ordinates of the vertices
int Y[ ] = { 1396, 1460, 1317, 1305 } ;
//Create a polygon object with the sides and the vertices
Pgon Math_Pgon ( nSides, X, Y ) ;
```

Figure 2.6: Code snippet illustrating Polygon creation for Math Bldg.

A visitor can be considered to be close enough to a particular building if his current location is within a specific distance from the building. If a distance of say 80 feet is used, then a visitor who is within 80 feet from any of the edges of the building is considered to be in the premises of the building. To facilitate this, every polygon in the Virtual Space must be fitted with an outer polygon that is a set distance away from the inner polygon. If the current x-y position of a visitor lies within one of the outer polygons, it means that he is in the premises of the building the polygon represents.

The Pgon class provides a method called 'Expand' that can be used to expand a polygon by a specified distance to obtain a new outer polygon. The code snippet in Figure 2.7 illustrates the creation of an outer polygon for the Math Building.

```
//Create a new polygon object
Polygon  Outer_Math_Pgon ;
//Specify the proximity distance
int nDistance = 80 ;
//Invoke the Expand method of Math_Pgon and pass the
//new object and the distance
Math_Pgon.Expand ( nDistance, Outer_Math_Pgon ) ;
```

Figure 2.7: Code snippet illustrating Outer Polygon creation

The Pgon class also provides a method called 'Contains' that can be used to find out if the current position of a visitor is inside or outside the outer polygon. The method takes the x and y values of the visitor's current position. It returns true if the point representing x-y lies inside the polygon; returns false otherwise. The code snippet in Figure 2.8 illustrates how the method is used.

```
//Set the x and y co-ordinates of the visitor
int currentX = 1400 ;
int currentY = 1450 ;
//Invoke the Contains method
if ( Outer_Math_Pgon.Contains ( currentX, currentY ) )
{
        //the visitor is in the premises of the Math Building
}
else
{
        //the visitor is not anywhere near the Math Building
}
```

Figure 2.8: Code snippet illustrating Contains method of Polygon class

Figure 2.9 summarizes the discussion in this section. It illustrates the Math Building in the Virtual Space including the outer polygon which is 60 feet away from the edges of the building. It shows the scenario where a visitor is away from the building and another scenario where the visitor is in the premises of the building.

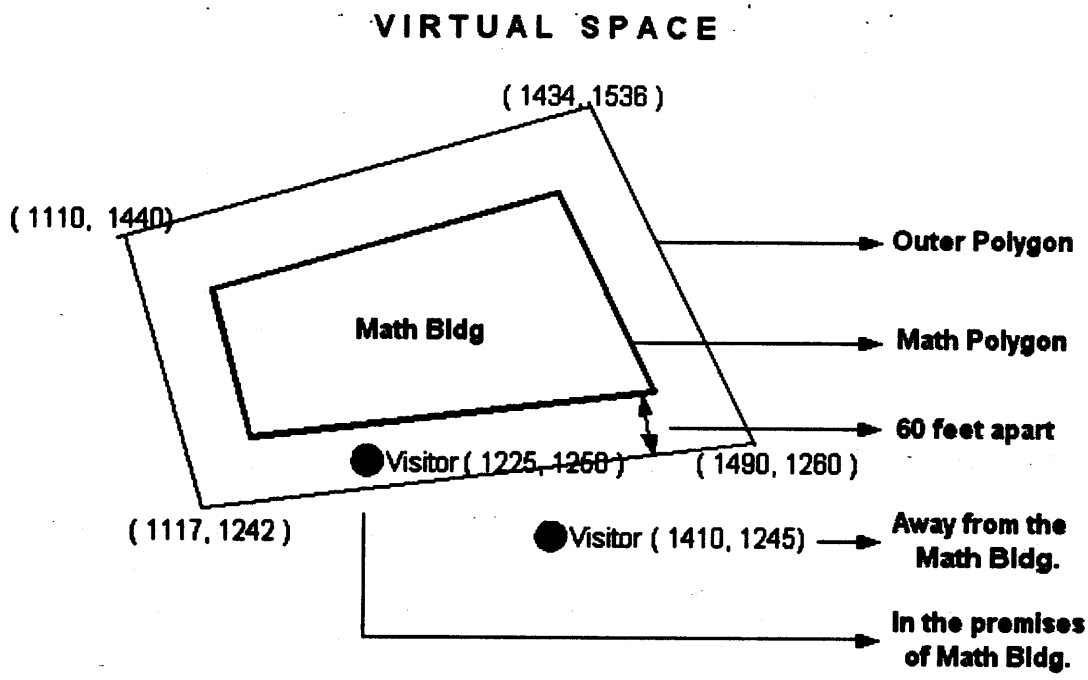

Figure 2.9: Math Building in the Virtual Space

As a proper foundation is crucial to the working of UbiTour, considerable amount of time and effort was spent on the implementation. The foundation was tested with different values for the proximity distance, i.e, the distance between the inner and the outer polygons, and an appropriate value was chosen. The accuracy of the GPS receiver should also be considered when selecting a distance value.

The next chapter provides a brief overview of the various services provided by UbiTour. The architecture, design and implementation issues are presented in later chapters.

CHAPTER 3

UBITOUR SERVICES

In this chapter a survey of some of the major features of the UbiTour architecture for electronic tourism is presented. Special attention is paid to the mapping of these features to 3G (or another type of network providing ubiquitous connectivity) and high-bandwidth WLAN. The architecture aims to provide a diverse set of services to visitors touring an area that has been UbiTour-"enabled". The prototype currently targets HP iPaq PDAs equipped with GPS receivers, though we are now looking at more powerful PDA platforms such as the Sony NZ90, which provides an integrated keyboard, higher resolution screen, and a high resolution digital camera.

The services envisioned are broadly divided into three categories: *Personalized services*, *Collaborative services* and *Broadcast-based services*. The first category includes services that are customizable for each visitor through a set of preferences, which are contained in a user's profile. Collaborative services facilitate communication among visitors, allowing them to share pictures, video, and comments, while tracking the locations of other visitors. The third category, broadcast-based services, enables dynamic information in the form of announcements and advertisements to be broadcast to visitors within a specific geographical area. Services in the first two categories are supported by other background services such as the Locator service, which keeps track of the visitors'

current locations, and the Group Manager service, which manages tourist groups. The next few sections examine representative services in each category.

3.1 Personalized Services

3.1.1 User Profile Service

The User Profile Service provides an applet that can be invoked by the visitor on the handheld unit to customize his or her user profile. The user profile is maintained remotely and is updated to reflect any changes made to it by the visitor. Some of the preferences that can be set in the user profile include the types of attractions the user is interested in visiting, options controlling multicast announcements, and options to control interaction with other users. As this service is responsible for storing contextual information, it will receive requests from other services that need such information. The User Profile service requires low bandwidth and ubiquitous connectivity; 3G connectivity can be used to support this service. The screenshots in Figure 3.1 illustrate the User Profile applet for a user who is interested in historical and architectural information, and wishes to receive advertisements and reveal his location to his group.

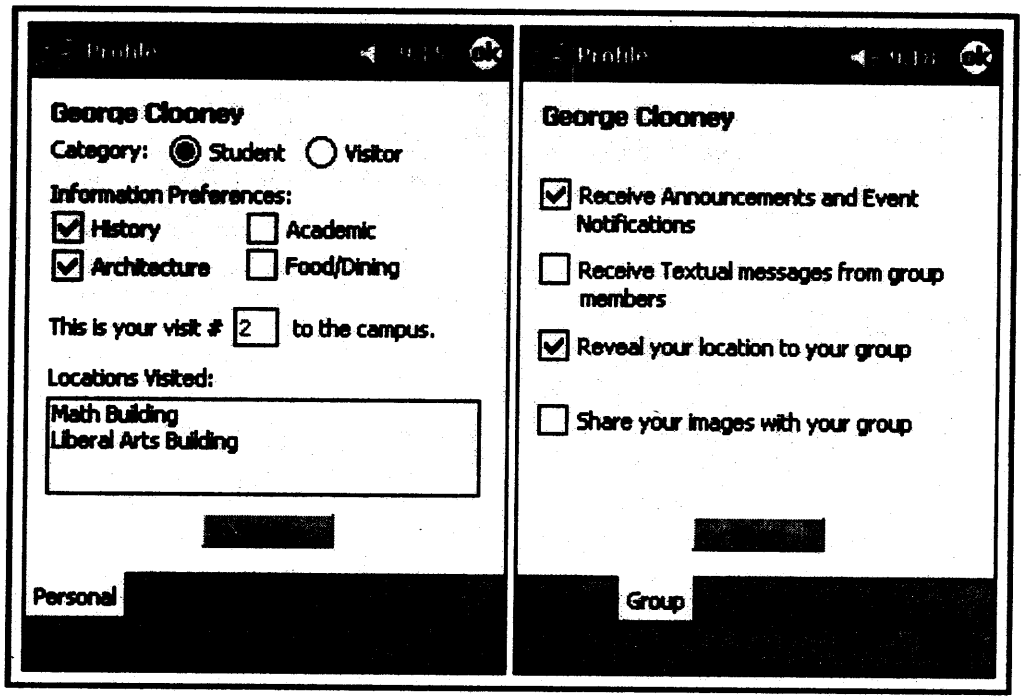

Figure 3.1: User Profile applet of a visitor

3.1.2 Information Service

The Information Service has access to the entire information base. This service enables visitors to view information about locations of interest based on their current position. The service first contacts the Locator service for the visitor's current location and then contacts the User Profile service to retrieve the visitor's personal preferences, allowing the user to see customized and location-based information. The service also provides visitors with images related to the site and associated web links. Connectivity for this service can be either through 3G or WLAN; WLAN is preferred, but can be used only if it is available and if information services can be reached (e.g., if the local WLAN provides broader Internet connectivity). The Figure 3.2 shows screenshots illustrating information about the UNO Mathematics Building for a visitor who is interested in academic issues.

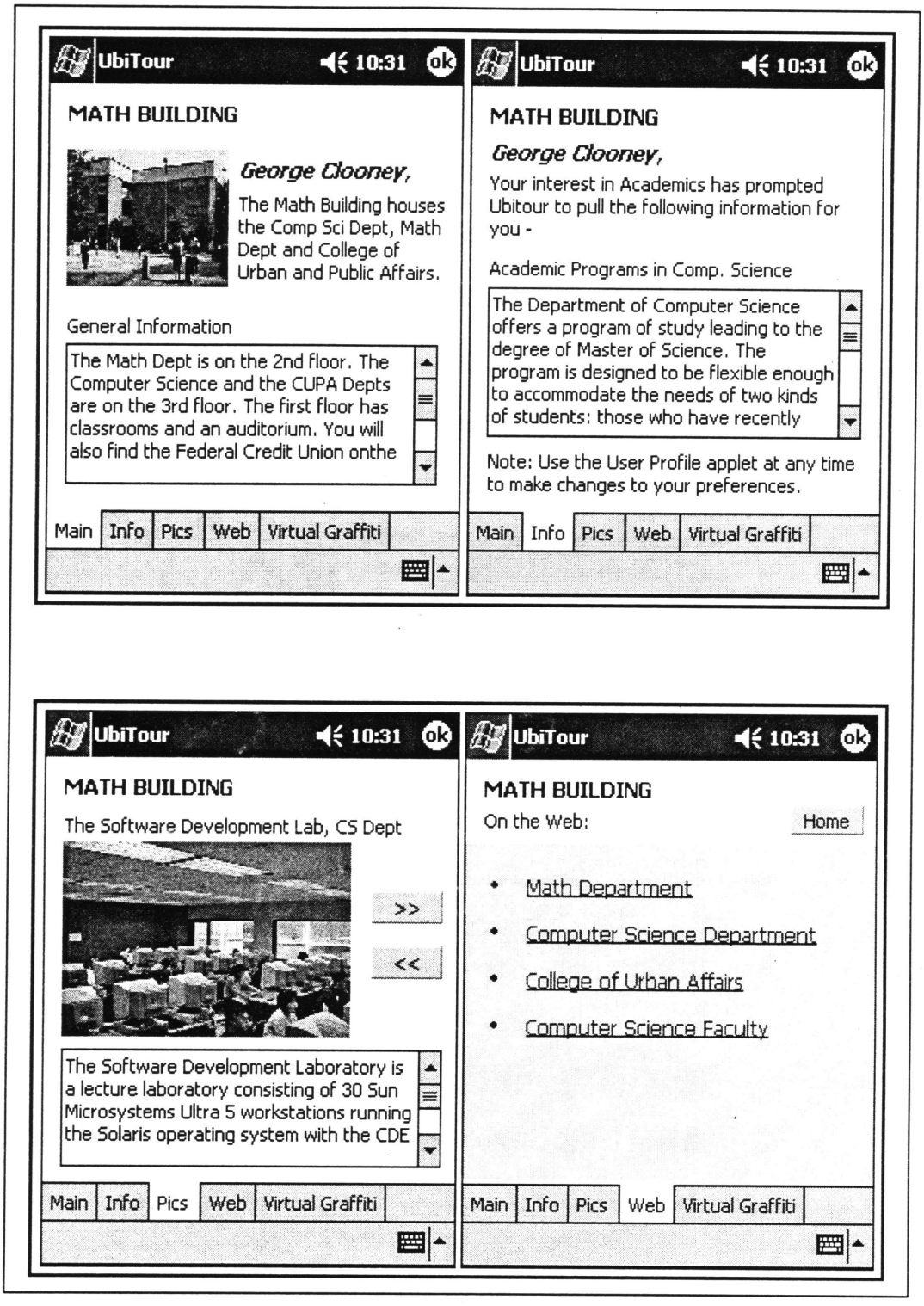

Figure 3.2: Information Pages for a visitor interested in Academics

3.1.3 Route Guidance Service

The Route Guidance service enables visitors to get directions to a specific destination. When a visitor indicates the desired destination, UbiTour calculates the route

from the visitor's current location to the indicated destination and delivers the complete directions to the visitor. The service also provides some minimal real-time route guidance. The Locator service is used to periodically retrieve the user's current location. To continuously track the visitor's movements and provide appropriate directions, 3G connectivity will be necessary and also sufficient for this service. Figure 3.3 below illustrates the directions provided to a visitor who is currently at the Math Building and wishes to visit the Liberal Arts Building next.

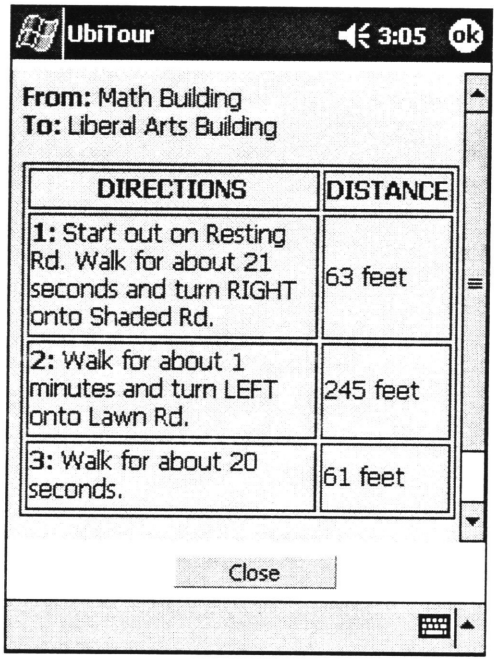

Figure 3.3: Directions from Math to Liberal Arts Building

3.1.4 Predefined Tours Service

This service will allow visitors to take predefined tours of the campus. This service is conceptual and hasn't been implemented. But with the basic infrastructure of UbiTour in place, development and deployment of this service should not be much of a

challenge. UbiTour can guide the visitors from one location of interest to another, allowing the visitors to spend time at each location. It can utilize the Information Service to provide tourist information and the Route Guidance service to provide directions from one destination to the next. 3G connectivity will be necessary for the route guidance part of the service, which requires continuous connectivity, and WLAN is preferred for the information services part, due to the high bandwidth needed.

3.1.5 Image & Video Transfer Service

With the introduction of PDAs such as the Palm OS-based Sony Clie PEG-NZ90, which tightly integrate a built in digital-camera that supports video recording and playback, two-way multimedia services will be easily supportable under the UbiTour architecture. The Image and Video Transfer Service will allow images and videos taken by the visitor, as he moves about, to be saved. A Web-based photo and video album can then be created automatically at the end of the tourist's visit. This service hasn't been implemented, primarily due to limitations of the hardware we have on hand; but the architecture of UbiTour allows easy inclusion of such services. WLAN connectivity is preferred, as high bandwidth is needed for swiftly off-loading video and images to a server. The off-loading could be done when pockets of WLAN connectivity are encountered, thereby freeing space on the mobile unit. Until then, the data can be cached on the PDA.

3.2 Collaborative Services

3.2.1 Virtual Graffiti Service

When a visitor arrives at a location of interest, this collaborative service allows the visitor to view comments made by previous visitors to the site. It also allows the visitor to post a comment about the location, which other visitors can view. This virtual

graffiti enables visitors to share their experiences and views, or leave contact information for other visitors (e.g., "Tony: Lost touch with you and I'm going back to the hotel to sleep for a bit. See you later."). As this service will be available only at locations of interest, where WLAN connectivity could be expected, WLAN can be used to provide this service. If WLAN connectivity is unavailable, 3G can be used as bandwidth requirements are low. Figure 3.4 illustrates virtual graffiti shown for Liberal Arts Building.

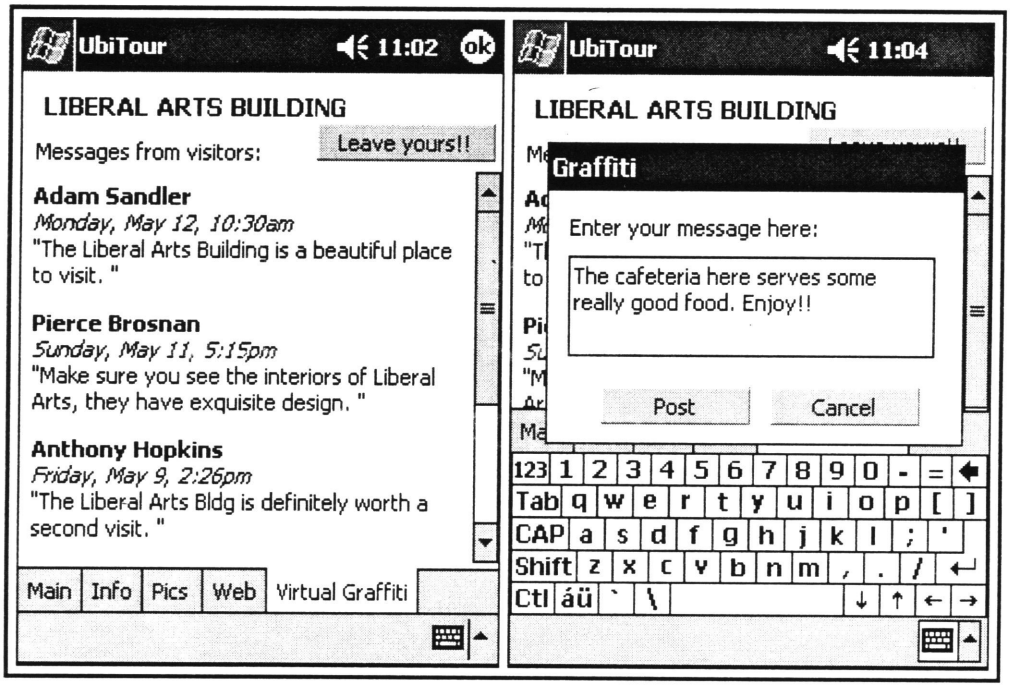

Figure 3.4: Virtual Graffiti for Liberal Arts Building

3.2.2 Location Sharing Service

The Location Sharing service enables a visitor to see the current locations of the other visitors in his group. If the current location of a visitor is unavailable, then the last known location of that visitor is used. To support privacy, the User Profile service gives

the visitor the option not to reveal his location to other visitors in his group. This necessitates the interaction of the Location Sharing Service with the User Profile Service. The service also interacts with the Group Manager and the Locator background services to determine the current locations of the group members. As this service requires ubiquitous connectivity and low bandwidth, 3G connectivity can be used to support this service. Figure 3.5 shows the current locations of the group a visitor belongs to.

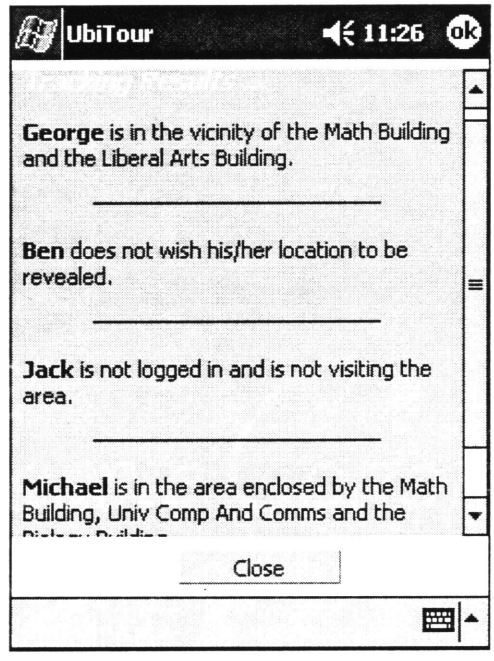

Figure 3.5: Current whereabouts of fellow group members

The page is refreshed at regular intervals to display up-to date locations of the group members.

3.2.3 Text Messaging Service

Text messaging can allow visitors who arrive in a group but intend to explore independently to remain in touch by sending simple text messages to each other. The service hasn't been implemented, but again, the architecture of UbiTour allows easy

inclusion of such a service. Taking into consideration a visitor's need for privacy, the User Profile service can provide the visitor with an option to turn off receiving messages from his group or from visitors outside his group. The Text Messaging service will need to interact with the Group Manager background service and the User Profile service to determine which group members do not wish to receive messages. To enable a visitor to receive messages when outside WLAN coverage, this service requires ubiquitous connectivity. Also, as low bandwidth is sufficient, 3G connectivity can be used to provide this service.

3.2.4 Images & Videos Sharing Service

A visitor may wish to share the pictures and videos he has taken with other visitors in his group before the end of the tour. This conceptual collaborative service can achieve the purpose by prompting the visitor to select the image or video to be shared, indicate the recipients and specify a short message. The selected image or video along with the message can then be made available to the indicated recipients for viewing. This service hasn't been implemented. This service will depend on the Group Manager background service and the Image & Video Transfer Service. This service requires high bandwidth, so WLAN connectivity is preferred for this service.

3.3 Broadcast – Based Services

3.3.1 Dynamic Information Delivery Service

The architecture supports the Dynamic Information Delivery service in order to make timely information available to visitors within a specific geographical area. This mechanism is based on UDP multicast, and specifically requires WLAN support, since there is no multicast architecture for 3G cellular networks. Any location of interest can utilize this mechanism by deploying an inexpensive 802.11 access point to disseminate

restaurant menus, advertisements, or the like. Internet connectivity for the access point supporting dynamic information delivery is not required, which helps keep infrastructure costs low and reduces security risks. If a visitor does not wish to receive notifications, he can utilize the User Profile service to turn off notifications. Figure 3.6 shows screenshots depicting the Computer Science Department sending information about an upcoming event to visitors currently in the vicinity of the building and the Biology Department notifying its visitors about early closing.

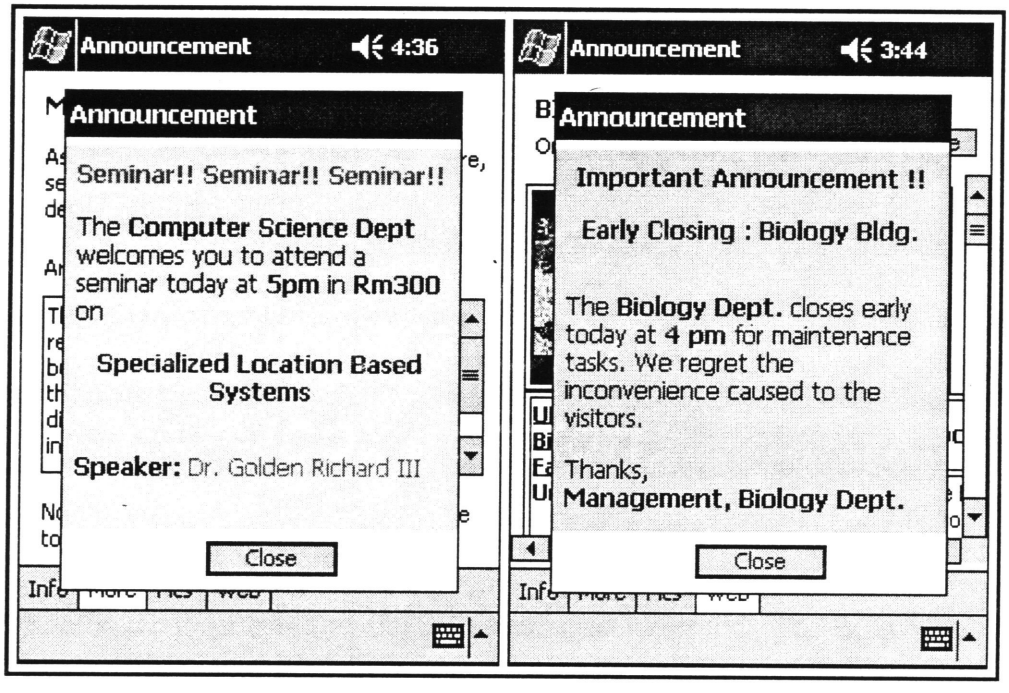

Figure 3.6: Announcements from Math and Biology Buildings

CHAPTER 4

UBITOUR ARCHITECTURE

UbiTour is a distributed, decentralized, and modular architecture. The primary benefits of such an architecture are scalability, elimination of single points of failure, and extensibility. The flexible integration of multiple wireless network technologies allows degraded service even in the presence of network failures (e.g., the loss of one or more WLANs at points of interest). SOAP (Simple Object Access Protocol) [5] is used as the primary communication protocol for the services to interact with each other and with the client units. SOAP enables services that are independently developed and deployed to interoperate with each other through well defined interfaces, thereby achieving a loosely-coupled, distributed and modular architecture.

4.1 UbiTour Service Architecture

Figure 4.1 illustrates the UbiTour service architecture. The SOAP interactions among the different services are shown. Service-specific data is stored in XML or other appropriate data structures. The background services are displayed in a lighter shade to differentiate them from the services provided to the users. Further, the specific type of connectivity used by each service (3G or WLAN) is indicated in the diagram.

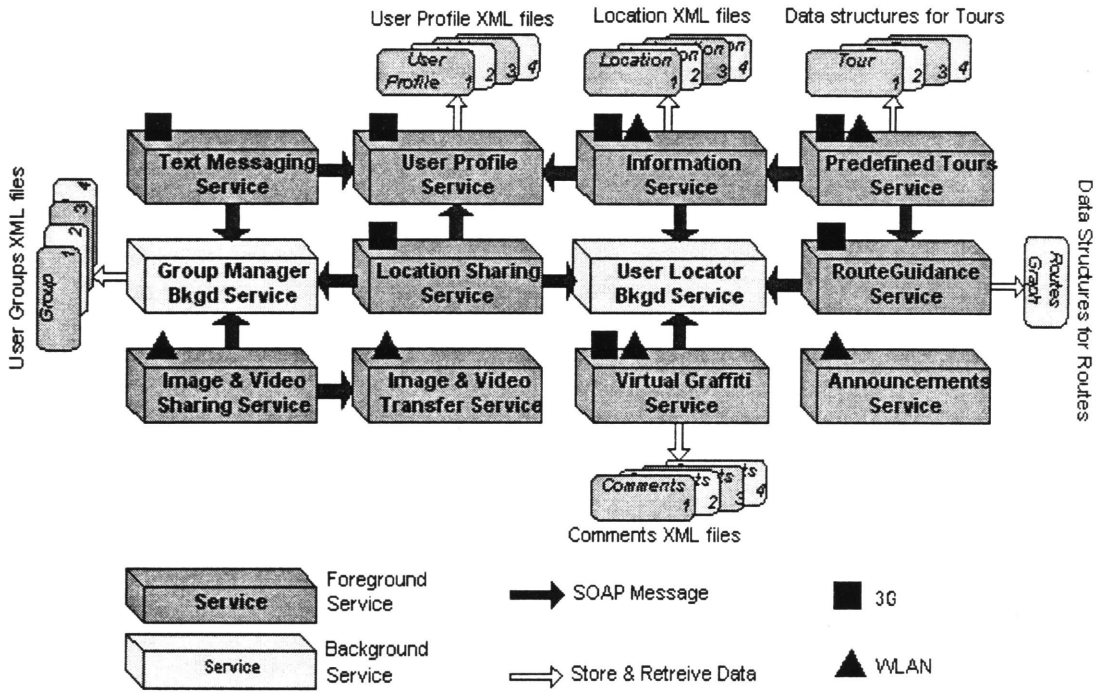

Figure 4.1: UbiTour Service Architecture

The different services interact with each other to provide the overall functionality of UbiTour. To get a general idea of the kind of interactions involved, a later subsection discusses the interactions associated with the Information and the Location Sharing services. The next subsection provides information on SOAP and gives sample SOAP request/reply messages occurring in UbiTour.

4.2 Communication using SOAP

SOAP is a lightweight protocol for the exchange of structured information between peers in a distributed and decentralized computing environment. The structured information is represented in XML. Though any underlying protocol can be used to carry SOAP messages, HTTP is the most prevalent one and almost all implementations of SOAP support HTTP binding. SOAP allows two models of information exchange; one

based on a request/reply scheme (i.e., Remote Procedure Call) and the other based on one-way messaging where one peer sends a message to another peer. The data types supported by SOAP include several simple data types such as integer, string, float, and complex data types such as records and arrays. Several implementations of SOAP are available today in different programming and scripting languages like Java, C++ and Perl. This combined with the interoperability offered by SOAP gives the developer the freedom to implement a service in a language of his choice and still make it interact with services written in other languages.

In UbiTour, the request/reply (Remote Procedure Call) model of information exchange is used and HTTP is used for carrying SOAP messages. The SOAP request parameters are embedded in an HTTP request message and the SOAP reply parameters are embedded in an HTTP response message. Figures 4.2 and 4.3 show the HTTP request and reply messages exchanged when the client calls the 'GetProfile()' remote method in the User Profile service to download the user profile of a visitor for displaying on the handheld. The GetProfile() method takes the ID of the user as an argument.

```
POST /axis/services/UserProfile HTTP/1.1
Host: mobile13.cs.uno.edu:8081
Accept-Charset: UTF-8, UTF-16;q=0.8, iso-8859-1;q=0.8
Accept-Encoding: deflate
Content-Type: text/xml; charset=UTF-8
SOAPAction: ""
User-Agent: PocketSOAP/1.4.3/RC3
Content-Length: 361

< S:Envelope    S:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
                xmlns:S=http://schemas.xmlsoap.org/soap/envelope/
                xmlns:XS=http://www.w3.org/2001/XMLSchema
                xmlns:XI=http://www.w3.org/2001/XMLSchema-instance
                xmlns:a="urn:UbiTour-userprofile" >
        < S:Body >
                < a:GetProfile >
                        < arg1 XI:type="XS:int" >1< /arg1 >
                < /a:GetProfile >
        < /S:Body >
< /S:Envelope >
```

Figure 4.2: SOAP Request for GetProfile( ) in User Profile Service

The ID of the user is contained in the 'arg1' element, which in turn is contained in the 'GetProfile' element.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Sun, 22 Jun 2003 03:11:47 GMT
Server: Apache Coyote/1.0

< ?xml version="1.0" encoding="UTF-8"? >
< soapenv:Envelope       xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
                         xmlns:xsd=http://www.w3.org/2001/XMLSchema
                         xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance" >
    < soapenv:Body >
        < ns1:GetProfileResponse  soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
                                  xmlns:ns1="urn:UbiTour-userprofile" >
            < GetProfileReturn xsi:type="ns2:Element" xmlns:ns2=http://xml.apache.org/xml-soap >
                < USERPROFILE >
                    < PROPERTYSET >
                        < PROPERTY >
                            < NAME >Name< /NAME >
                            < VALUE >George Clooney< /VALUE >
                        < /PROPERTY >
                        < PROPERTY >
                            < NAME >InfoPref< /NAME >
                            < VALUE >UB_INFOCATEGORYTYPE_ACADEMIC:< /VALUE >
                        < /PROPERTY >
                        .....
                        .....
                        < PROPERTY >
                            < NAME >LocationReveal< /NAME >
                            < VALUE >true< /VALUE >
                        < /PROPERTY >
                    < /PROPERTYSET >
                < /USERPROFILE >
            < /GetProfileReturn >
        < /ns1:GetProfileResponse >
    < /soapenv:Body >
< /soapenv:Envelope >
```

Figure 4.3: SOAP Reply for GetProfile( ) in User Profile Service

The entire user profile is embedded in the 'GetProfileReturn' element as a set of 'Property' elements. Each Property element consists of the name of the property and the value. The client application parses the XML to extract the user profile and then displays it.

All interactions between the client and the services and also any interactions among the services themselves is based on a similar HTTP request/reply scheme where

the SOAP payload is embedded in an HTTP message. The only service that does not fit into the SOAP paradigm is the Dynamic Information Delivery service, which is based on UDP Multicasting. More information on that service is provided in a later chapter.

4.3 Service Interactions

The overall functionality of UbiTour is achieved by the several interactions taking placing among the different services, some of which are pretty involved, while others quite simple. A proper understanding of these interactions is necessary in order to grasp the distributed and decentralized nature of working that is the basis of UbiTour. In this section, I break open the Service Architecture diagram and dwell on two specific pieces, one, the interactions associated with the Information Service and two, the interactions associated with the Location Sharing Service.

4.3.1 Information Service Interactions

The Information Service needs to interact with the Locator background service and the User Profile Service to provide location – based and customized information to the client. The Service Interaction Diagram for the Information Service is shown in Figure 4.4 below. Each box represents a service. Along with the service name, the box also shows the name of the method that is invoked remotely by another service or client. All the SOAP request/response messages are numbered that indicate the order in which they occur.

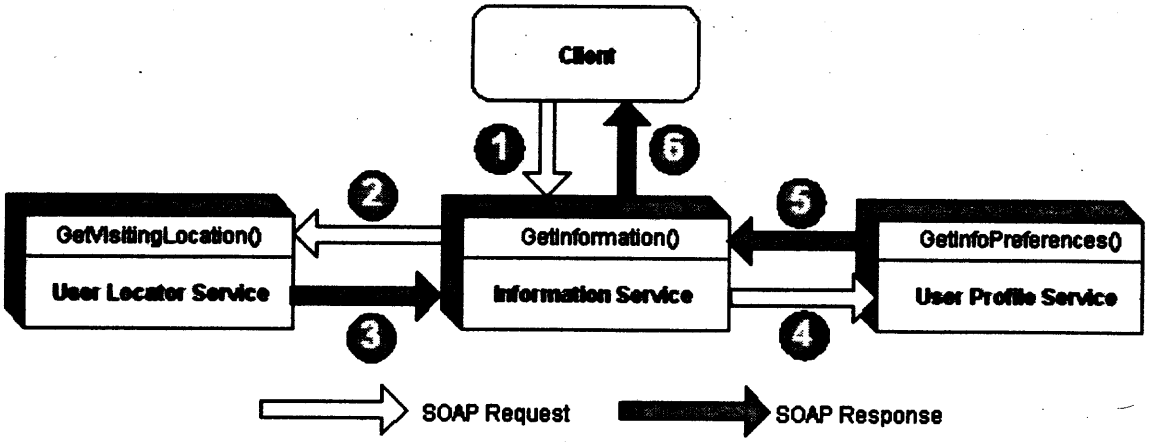

Figure 4.4: Information Service Interactions

The figure is self explanatory. The client invokes the GetInformation( ) method in the Information Service. The method then invokes the GetVisitingLocation( ) method in the User Locator Service to get the current location of the user. The GetInfoPreferences( ) method of the User Profile Service is then invoked to get the information preferences of the user. The service can then customize the information for the current location based on the extracted preferences and send the information back to the client.

4.3.2 Location Sharing Service Interactions

The Location Sharing Service needs to interact with the Locator and Group Manager background services and the User Profile service to enable a client to track the locations of his group members. The Service Interaction Diagram for the Location Sharing Service is shown below, followed by a brief explanation of each SOAP request/reply involved.

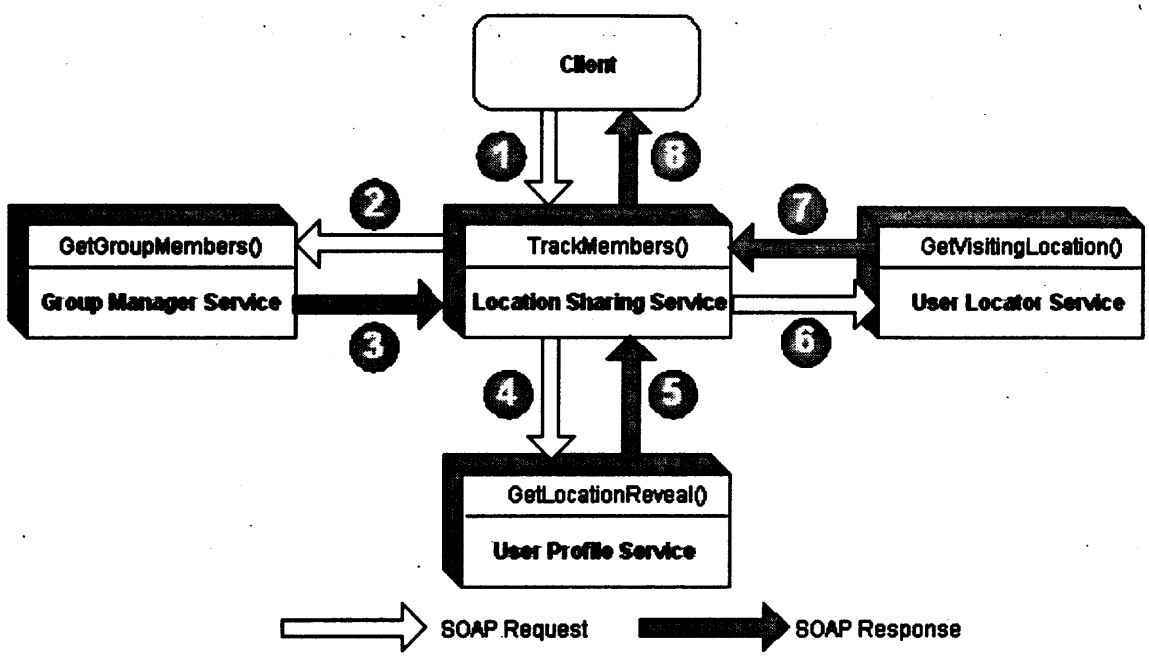

Figure 4.5: Location Sharing Service Interactions

As seen in the figure, the client invokes the TrackMembers( ) method in the Location Sharing Service. The method first invokes the GetGroupMembers( ) method in the Group Manager Service to get the group members of the user. The GetLocationReveal( ) method in the User Profile Service is then invoked to determine which users wish to reveal their location. The User Locator Service is then contacted to get the current locations of all the members who wish to reveal their location. The current locations are sent back to the client for display.

The interactions associated with the Information and the Location Sharing services are representative of the interactions that occur for all the other services and give a general understanding of the basic architecture of UbiTour. In the next chapter, the design and implementation issues are discussed which provide an in-depth understanding of the working of UbiTour.

CHAPTER 5

DESIGN AND IMPLEMENTATION

Equipped with a general understanding of the architecture of UbiTour, a discussion of the design and implementation of the UbiTour services and the client side application will provide an in-depth view of the working of the various services and their interaction with each other and with the client side application.

All the UbiTour services, except the User Locator Service, the Route Guidance Service and the Dynamic Information Delivery Service have been implemented in Java using Apache Axis [8] for SOAP communication. Apache Axis is a Java implementation of the SOAP protocol. The HTTP binding is provided by embedding Axis in the Apache TOMCAT Web Server. The Xerces Java parser is used wherever XML parsing is required. The User Locator Service and the Route Guidance Service have been implemented in C++ using gSOAP [9] for SOAP communication. gSOAP is a C++ SOAP implementation that can run as a standalone SOAP server or can be integrated into Microsoft Internet Information Server (IIS). For this service, the Xerces C++ parser is used for XML parsing. The Dynamic Information Delivery Service has also been implemented in C++ to utilize the UDP multicasting functions.

The client side application has been written in Embedded Visual C++ and all the user interface screens have been developed using MFC (Microsoft Foundation Classes). A COM (Component Object Model) client for SOAP called PocketSOAP [10] has been

used for making SOAP calls from the client application. A C++ set of classes called VOImage [11] has been used for displaying images on the handheld. Another popular set of C++ classes called STHtmlDialog [12] has been used for incorporating HTML into MFC dialogs.

Section 5.1 discusses the design and implementation issues of UbiTour services. Section 5.2 provides a brief overview of the design of the client side application.

5.1 UbiTour Services

Chapter 3 provided a general overview of all the services provided by UbiTour. In this section, the services are looked at more deeply from a design and implementation point of view.

5.1.1 User Profile Service

The User Profile Service allows visitors to customize their user profile using the User Profile applet on their handheld. The user profile of every visitor is stored remotely as an XML file and is updated by the service whenever any changes are made to it on the client side by the visitor. The user profile of a visitor contains information like the name of the visitor and locations visited so far. The user profile also contains preferences that can be set by the visitor. There are personal preferences and group preferences. Personal preferences include the categories of information the user is interested in such as history and architecture. Group preferences include options for turning on and off announcements and event notifications, receiving messages from group, revealing location and sharing images with group members.

The preferences and other information are stored in an XML file that is accessible to the User Profile service. A sample structure of such an XML file is shown in Figure 5.1. The UID and GID tags contain the Id of the user and of the group he belongs to

respectively. The LOCATIONSVISITED tag contains the Ids of the locations the user has so far visited. The PERSONALPREF tag has two child tags, UCATEGORY that indicates the user category the visitor belongs to, and the INFOPREF tag which indicates his information preferences. The group preferences are stored in the GROUPPREF tag, for example, the EVENTSRECV tag indicates whether the user is interested in receiving announcement.

```
< UPROFILE >
        < UID >1< /UID >
        < GID >1< /GID >
        < NAME >
                < FIRST >George< /FIRST >
                < LAST >Clooney< /LAST >
        < /NAME >
        < VISITSCOUNT >1< /VISITSCOUNT >
        < LOCATIONSVISITED >101:102:103< /LOCATIONSVISITED >
        < PERSONALPREF >
                < UCATEGORY >STUDENT< /UCATEGORY >
                < INFOPREF >UB_INFOCATEGORYTYPE_HISTORY< /INFOPREF>
        < /PERSONALPREF >
        < GROUPPREF >
                < EVENTSRECV >FALSE< /EVENTSRECV >
                < MESSAGESRECV >TRUE< /MESSAGESRECV >
                < LOCREVEAL >FALSE< /LOCREVEAL >
                < IMAGESHARE >TRUE< /IMAGESHARE >
        < /GROUPPREF >
< /UPROFILE >
```

Figure 5.1: User Profile XML File

The information currently stored in the profile is only a subset of what would be stored in a production version of the system. To facilitate easy and effortless inclusion of new preferences for all visitors and removal of existing preferences, all the allowed personal and group preferences are stored in an XML file along with their name and the XPath expression to retrieve the associated values from user profile XML files. A single instance of this file is maintained by the service and its structure is as shown in Figure

5.2. The NAME tag contains the name of the preference and the ACCESSPATH tag contains the XPATH expression to the preference in an actual profile file.

```
< TARGETLIST >
        < TARGET >
                < NAME >InfoPref< /NAME >
                < ACCESSPATH >/UPROFILE/PERSONALPREF/INFOPREF< /ACCESSPATH >
        < /TARGET >
        < TARGET >
                < NAME >Announcements< /NAME >
                < ACCESSPATH >/UPROFILE/GROUPPREF/EVENTSRECV< /ACCESSPATH >
        < /TARGET >
        < TARGET >
                < NAME >Messaging< /NAME >
                < ACCESSPATH>/UPROFILE/GROUPPREF/MESSAGESRECV</ACCESSPATH >
        < /TARGET >
        < TARGET >
                < NAME >LocationReveal< /NAME >
                < ACCESSPATH >UPROFILE/GROUPPREF/LOCREVEAL< /ACCESSPATH >
                  < /TARGET >
        < TARGET >
                < NAME >ImageSharing< /NAME >
                < ACCESSPATH >/UPROFILE/GROUPPREF/IMAGESHARE< /ACCESSPATH >
        < /TARGET >
< /TARGETLIST >
```

Figure 5.2: XML file containing Preference Names and XPATH Expressions

When a new preference needs to be incorporated, it must be added to the list of preferences in the above XML file. When the service is first deployed and initialized, it picks up all the preferences from this XML file and thus knows what are the preferences provided to the visitors and how to access them using the XPath expressions.

The service exposes a set of methods that can be invoked remotely by clients or other services. Some of the significant methods are explained below.

- **LoadProfile( )**

When a visitor first logs into the UbiTour system, the client calls the LoadProfile() method which takes the ID of the user. The method reads the appropriate

user profile XML file, extracts all the information and stores it in a Java HashMap object with the name of the preference as the key. This allows faster access to the preference values whenever they need to be read or modified later.

- **GetProfile( )**

After loading the profile remotely, the client calls the GetProfile( ) method which returns the entire profile of the user as an XML DOM Element. The client parses the returned XML element and extracts all the user profile settings so that they can be displayed on the handheld whenever requested by the user.

- **UnloadProfile( )**

When a visitor logouts from UbiTour, the client calls the UnloadProfile(method. The method writes the current state of the associated HashMap object back to the user profile XML file and removes the HashMap object from memory. This facilitates persistence of preference settings across logins.

The service also provides methods to get and set preference values like GetInfoPreferences( ), TurnMessagingOn( ), TurnMessagingOff( ), TurnLocationRevealOn( ) and TurnLocationRevealOff( ). This service receives requests from other services that require the preferential settings of the visitors.

5.1.2 Information Service

The Information Service provides location – based and customized information to visitors. The information provided by this service is not only dependent on the visitor's current location but also customized to suit the personal preferences of the visitor.

The service has access to the entire information base which consists of files of information associated with all the locations of interest. The information could be just plain text, images or web links. Every piece of information, be it text or image or a web

link, is associated with a category of information. As an example, a piece of text that describes the history of the Math Building belongs to the 'historical' category; whereas an image that shows the labs in Computer Science Department will belong to the 'academic' category. This categorization facilitates information to be filtered based on a visitor's preferences thereby resulting in customized information. A visitor who indicates interest in architecture will be sent pieces of information that belong to the architecture category. A general category is also included and information that belongs to this category is sent to the visitor irrespective of his preferences. Such a category is required to represent information like welcome messages, introductory text, etc, that every visitor must see.

An important requirement of this service is to be able to modify the information base without the need to make any changes to the service code. The information base will be modified once in a while whenever new information is available and needs to be included or when information that is stale or nor valid anymore needs to be removed from the information base.

Every location of interest has an associated XML file which has an entry for every piece of information. Each entry consists of the category the piece of information belongs to and the name of a file that contains a description of the piece of information including the actual filename if it's an image or a text file. As an example, the XML file for the Math Building is as shown in Figure 5.3:

```
<UBINFORMATION>
    <AID>101</AID>
    <INFOSET>
        <INFO>
            <TYPE>UB_INFOTYPE_TEXT</TYPE>
            <CATEGORY>UB_INFOCATEGORYTYPE_GENERAL</CATEGORY>
            <FILE>IF101UBINFO2.XML</FILE>
        </INFO>
        <INFO>
            <TYPE>UB_INFOTYPE_TEXT</TYPE>
            <CATEGORY>UB_INFOCATEGORYTYPE_HISTORY</CATEGORY>
            <FILE>IF101UBINFO3.XML</FILE>
        </INFO>
        <INFO>
            <TYPE>UB_INFOTYPE_IMAGE</TYPE>
            <CATEGORY>UB_INFOCATEGORYTYPE_ACADEMIC</CATEGORY>
            <FILE>IF101UBINFO7.XML</FILE>
        </INFO>
        .....
        .....
    </INFOSET>
</UBINFORMATION>
```

Figure 5.3:: Information XML File for Math Building

When a new piece of information needs to be included for the Math Building, an entry must be added for it in the above XML file. When a piece of information needs to be removed, its corresponding entry must be deleted from the above XML file. When the Information service needs to load the information for a specific location of interest, it reads the XML file associated with that location of interest and extracts all the pieces of information and their categories. This makes the service code unaffected by changes made to the information base. Any change only requires the appropriate XML files to be modified.

The only significant method exposed by this service is the GetInformation( ) method which is invoked remotely by a client. The method when invoked contacts the User Locator Service for the visitor's current location, extracts the information for the current location, contacts the User Profile Service for the visitor's preferences, filters the

extracted information based on the preferences and sends customized information back to the client for display.

The GetInformation( ) method handles the condition where the visitor is in the vicinity of more than one location of interest. When such a situation arises, the method extracts the name and an image of every location of interest in the vicinity and sends them back to the client indicating that location resolution is required. The client then displays the images received along with their names and prompts the visitor to select the one he is actually looking at. The information for the selected location is then downloaded and displayed. The screenshot below illustrates this situation where the visitor is in the vicinity of both the Math and the Liberal Arts Building:

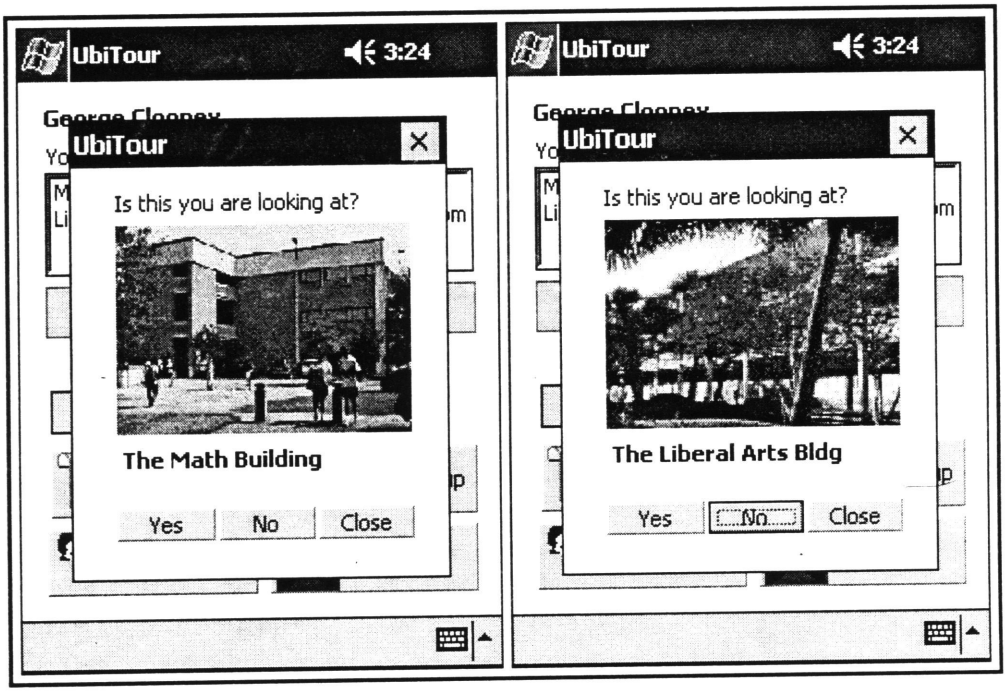

Figure 5.4: Location Resolution for Math and Liberal Arts Buildings

The Information Service exposes other helper methods like GetLocationNames( ) which takes an array of location IDs and sends back an array of strings containing the location names.

5.1.3 Route Guidance Service

The Route Guidance service provides directions to a visitor to go from one location of interest to another. C++ was chosen as the language for implementation to utilize the ProxUtils C++ library.

A graph of the entire area is first obtained by determining all the major routes and points. The points include locations of interest as well as route intersections. The routes become the edges of the graph and the points become the vertices. Dijkstra's shortest path algorithm has been used to calculate the route from one vertex to another. The algorithm only gives the vertices that form the shortest route. A visitor would also need information such as the turns he needs to take at the intersections. To determine the turns, Vector algebra has been used. The ProxUtils library has a Vector2D class that provides common vector calculations, specifically cross product. By considering two adjacent edges as vectors, the cross product of the two vectors indicates the turn a visitor will have to make while moving from the first edge to the second.

The significant method exposed by this service is the ProvideRouteGuidance( ) method. It takes the Id of the user making the request, the source and the destination locations. It calculates the route as explained above, prepares an HTML page with proper directions and distances and sends it back to the client.

5.1.4 Virtual Graffiti Service

When a visitor arrives at a location of interest, the Virtual Graffiti service enables the visitor to view comments made by other visitors who previously visited the same location. The service also enables the visitor to post a comment about the location which other visitors can then read.

The implementation of this service is rather straightforward. It exposes two specific methods, GetGraffiti( ) to retrieve the comments made for a specific location, and AddComment( ) to append the comment made by a visitor to the list of comments for a specific location. The comments are all stored in XML files, one file for every location of interest. The name of the visitor and the date and time at which he made the comment are also stored along with the comment itself.

When the service is initialized, information from all the XML files is extracted and stored in a Java HashMap object with the location ID as the key. This facilitates faster access of information for the GetGraffiti( ) method. When a visitor posts a comment, the AddComment( ) method is invoked remotely with the name of the visitor, the comment and the location. The method then appends this information along with the current date and time to the list of comments for the location. When the service is closed, all the comments for all the locations are written back to their appropriate XML files.

5.1.5 Location Sharing Service

The Location Sharing Service enables a visitor to track the locations of his group members. As described in Section 4.3.2, the TrackMembers ( ) method of this service needs to interact with the User Locator, Group Manager and the User Profile services. The service does not store any state information. It handles cases where if a particular group member is not logged into the UbiTour system, it sends an appropriate message about the member to the calling client.

5.1.6 Dynamic Information Delivery Service

The Dynamic Information Delivery Service is based on UDP Multicast that allows information to be delivered over the wireless network infrastructure to visitors who are within a specific geographical distance from the source of the information

delivery. The two C++ libraries that support this service are the DIDSrvr, which multicasts UDP packets containing information to be delivered and the DIDClient, which listens on the client for UDP multicasts.

The libraries have been designed to support the dynamic delivery of messages from both static as well as dynamic locations of interest. Examples of static locations include buildings, parking lots, etc. All such static locations have fixed IDs and include their IDs in the message. On the client side, the ID included in the received message is checked against the ID of the visitor's current location. If they match, the message is displayed; otherwise it is ignored. Dynamic locations of interest can pop up anywhere in an area. For example, a portable coffee shop might advertise its availability to visitors, inviting them to have a cappuccino. To facilitate this, the DIDSrvr library provides a mechanism wherein the GPS location of the stand and the distance of propagation can be delivered along with the message. When the client side module receives the message, it checks to see if the visitor's current location is within the propagation distance and if so, displays the advertisement; otherwise, the message is ignored.

A fixed multicast IP address of '234.5.6.7' and the port 7815 are used. The code snippet in Figure 5.5 shows how the DIDSrvr library can be used to deliver information from a static location:

```
//Create a DIDSrvr object
DIDSrvr   Server ;
//Initialize the DIDSrvr object with the multicast IP address and port
Server.Init ( "234.5.6.7", 7815 ) ;
//Create a DIDHdr object with the ID of the static location
DIDHdr   Header ( "LOC101" ) ;
//Create a DIDMsg object with the text message to be delivered
DIDMsg   Message ( "This is a message from a static location" ) ;
//Create a DIDPkt object with the DIDHdr and the DIDMsg objects
DIDPkt   Packet ( &Header, &Message ) ;
//Send the DIDPkt object
Server.Send ( &Pkt )
```

Figure 5.5: Usage of DIDSrvr Library

The code snippet in Figure 5.6 illustrates how a dynamic location can deliver a message.

```
//Create a DIDSrvr object
DIDSrvr   Server ;
//Initialize the DIDSrvr object with the multicast IP address and port
Server.Init ( "234.5.6.7", 7815 ) ;
//Create a DIDHdr object with latitude and longitude values of
// the location and the distance of //propagation
DIDHdr   Header ( 30.25, NORTH, 90.18, WEST, 1000, FEET ) ;
//Create a DIDMsg object with the text message to be delivered
DIDMsg   Message ( "This is a message from a dynamic location" ) ;
//Create a DIDPkt object with the DIDHdr and the DIDMsg objects
DIDPkt   Packet ( &Header, &Message ) ;
//Send the DIDPkt object
Server.Send ( &Pkt )
```

Figure 5.6: Message Delivery by a Dynamic Location

As can be seen from the above two code snippets, the only difference between a static and dynamic location is in the creation of the header (DIDHdr) object. For a static location, the object is created with the ID of the location; whereas for a dynamic location, the header object is created with the latitude and longitude values of the location, and the propagation distance.

The DIDClient library consists of the DIDClient and the DIDHandler classes. An object of the DIDClient class must first be created and its Init method should be invoked with the multicast IP address and port to perform initialization. The DIDHandler class provides 2 methods – HandleDecisionParams( ) to handle the header content and HandleMessage( ) to handle the actual message. These methods are called by the DIDClient object when a message arrives. The default implementations of these two methods do nothing. The user of the library must derive a class from the DIDHandler class and override the two methods. This enables the user to provide his own implementation of the handling of the header and the actual message. The DIDHandler class provides other useful methods like DIDCompareID( ) to compare the IDs of 2 static locations and DIDWithinRange( ) to check if the current location of the visitor falls within the propagation distance of the message.

5.1.7 User Locator and Group Manager Background Services

The User Locator and the Group Manager Services are background services that maintain essential information about visitors and receive requests from other services that need the information. They do not provide any visible service to the visitors. The User Locator service keeps track of the current locations of the visitors. The Group Manager service keeps track of visitor groups.

The User Locator Service has access to the Virtual Space and utilizes the ProxUtils C++ library to keep track of the proximity of visitors to the locations of interest in the Virtual Space. The service periodically receives location updates from clients. The two significant methods exposed by this service are:

- **UpdateLocationPhy( )**

Every UbiTour client periodically invokes this method and passes the latitude and longitude values obtained from the GPS receiver. The method maps the physical position into the Virtual Space and updates the proximity information of the visitor.

- **GetVisitingLocation( )**

This method is invoked by other services when they need to know the location/s a visitor is currently at.

The only significant method exposed by the Group Manager Service is the GetGroupMembers( ) method. When invoked, this method returns the members of the group the visitor belongs to. This service receives requests from the collaborative services for group information.

5.2 UbiTour Client Application

The UbiTour client application that runs on the handheld is designed as a set of light-weight modules, some of which interact with the UbiTour services, while others deal with local processing. The modules are listed as follows:

- **Control Module:**

This is the central module that initializes all the other modules and co-ordinates their activities. Every module has a reference to the Control module and all communication among modules takes place through this module.

- **Positioning Module:**

This module is responsible for periodically polling the GPS receiver, parsing the acquired data for location information and updating the User Locator Service with the new location.

- **GUI Module:**

The GUI Module manages the display of the various user interface screens.

- **UserProfile Module:**

This module is responsible for communicating with the User Profile service for retrieving and updating preferences and other information associated with the profile.

- **Information Module:**

This module is responsible for communicating with the Information service to retrieve information about the location the visitor is interested in.

- **VirtualGraffiti Module:**

This module is responsible for communicating with the Virtual Graffiti service to extract comments made for a specific location of interest and also to upload comments made by the visitor.

- **Announcements Module:**

This module utilizes the DIDClient library to listen for incoming messages from the Announcements service.

- **LocationSharing Module:**

This module is responsible for communicating with the Location Sharing service to track the locations of group members. The module periodically resends the tracking request to receive updated information.

Figure 5.7 illustrates the client modules and their interactions among themselves and with the services.

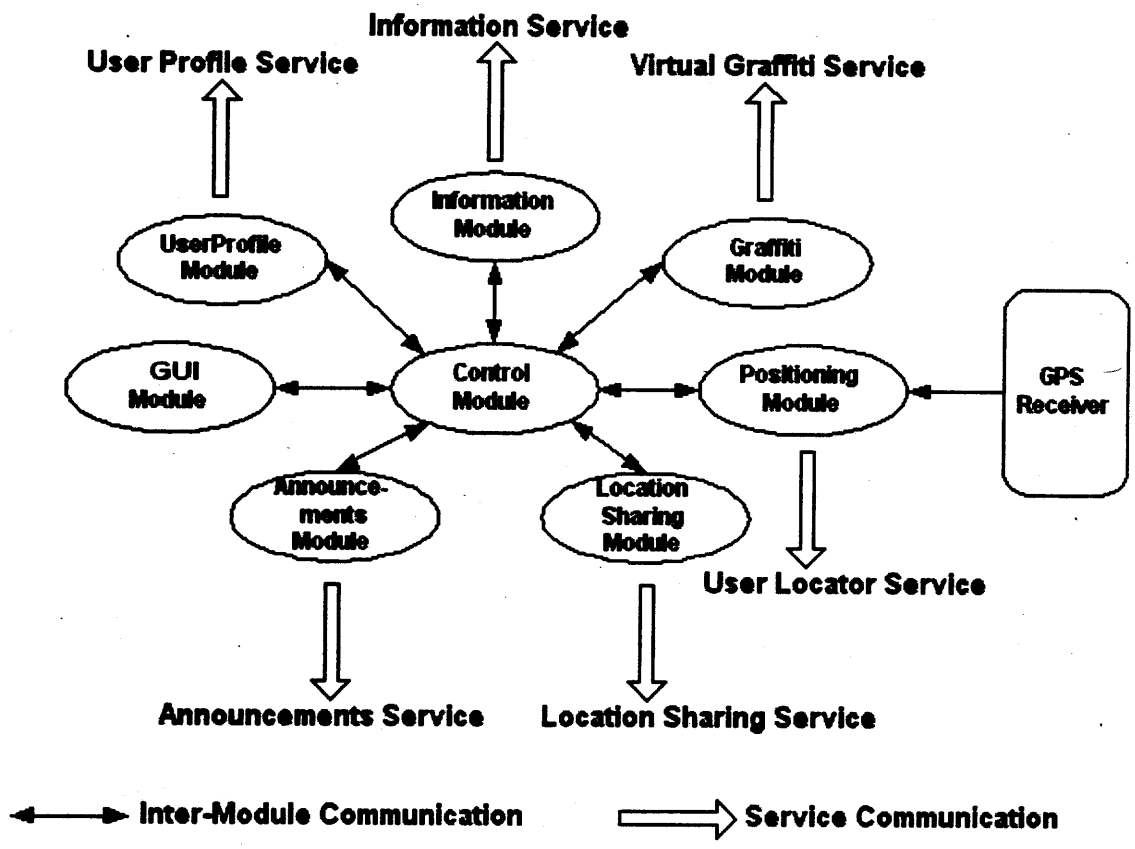

Figure 5.7: UbiTour Client Modules

As mentioned earlier, the modules that need to communicate with the services use PocketSOAP which is a COM SOAP client component.

This concludes the discussion of the design and implementation issues of UbiTour services and the client. Though SOAP is meant for easy interoperability, some effort was required in making the three different SOAP implementations, Apache Axis, gSOAP and PocketSOAP to communicate with each other, especially when a user defined object needs to be transferred from one service to another.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this report, the UbiTour architecture to support electronic tourism has been presented. The novel characteristic of the architecture is the intelligent integration of 3G and WLAN, attempting to derive maximum benefit for the user from each wireless technology. WLANs based on 802.11 offer high bandwidth, but ubiquitous deployment is unlikely. 3G wireless can be used to bridge pockets of 802.11-based WLANs, allowing ubiquitous connectivity, but does not currently provide bandwidth as high as WLAN.

The services offered by UbiTour have varying requirements in terms of bandwidth and connectivity and some prefer one type of connectivity over another. For example, services like Route Guidance, Text Messaging and Location Sharing require 3G connectivity for proper operation. The Image and Video Transfer and Sharing services will have better performance over WLAN, but can operate in a degraded fashion over 3G. Services like Dynamic Information Delivery, which are inherently broadcast-based, require WLAN for proper operation.

The modular nature of the architecture allows new services that exploit the benefits of both 3G and WLAN connectivity to be developed and deployed. Using SOAP, with HTTP as the transport mechanism, provides access to external Web-based services that might be relevant to electronic tourism. For example, in a commercial

deployment a weather service providing up-to-date weather forecasts might be made available to tourists. If such a service requires a fee, the tourism authorities could pay for the service and provide it free of charge to the tourists. A wide range of electronic tourism services with different bandwidth, connectivity and latency requirements can be imagined and the UbiTour system serves as an ideal testbed for evaluating the integration of these services over 3G and WLAN networks. This integration is made possible by the distributed and decentralized architecture of UbiTour.

A prototype implementation of the architecture currently runs on HP iPaqs. A major future work would be to expand the implementation to include the Sony PEG-NZ90, which has an integrated high-resolution digital camera. This will properly support the Image and Video Sharing and Transfer services. Also included as part of the future work is the implementation of some of the services which are still conceptual. But with a basic infrastructure in place, implementation and inclusion of new services shouldn't be much of a hassle. Finally, Bluetooth support is also being considered, to allow interaction with Bluetooth-based services in the environment and with Bluetooth devices owned by the tourist.

BIBLIOGRAPHY

[1] K. Cheverst, N. Davies, K. Mitchell, A. Friday, Experiences of Developing and deploying a Context-Aware Tourist Guide: The GUIDE Project, in Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, August 2000.

[2] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, M. Pinkerton, Cyberguide: A Mobile Context-Aware Tour Guide. Wireless Networks October 1997, Volume 3 Issue 5.

[3] W. Chan, Project Voyager: Building an Internet Presence for People, Places, and Things. Masters Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. (May 18, 2001).

[4] R. Mark, Standards from IEEE 802 Unleash the Wireless Internet, IEEE Microwave Magazine, pp.46–56, June 2001.

[5] Simple Object Access Protocol (SOAP) 1.1 W3C Specification http://www.w3.org/TR/SOAP/

[6] J. Haartsen, M. Naghshineh, J. Inouye, O. J. Joeressen, W. Allen, Bluetooth: Vision, Goals, and Architecture," Mobile Computing and Communications Review, October 1998, Volume 2 Issue 4.

[7] The National Marine Electronics Association, http://www.nmea.org

[8] Apache Axis, JAVA SOAP Implementation, http://ws.apache.org/axis/

[9] gSOAP, C++ SOAP Implementation, http://www.cs.fsu.edu/~engelen/soap.html

[10] PocketSOAP, COM Client Implementation for PocketPC http://www.pocketsoap.com/pocketsoap/

[11] CVOImage, Image Utility Class for PocketPC http://www.pocketsoap.com/pocketsoap/

[12] STHtmlDialog, HTML Utility Library for PocketPC http://www.pocketpcdn.com/articles/htmldialog.html

VITA

Vivek Chinta was born in Hyderabad, India in August, 1978. He graduated from Chaitanya Bharathi Institute of Technology, Hyderabad, with a Bachelor of Engineering in Computer Science And Engineering.

In August 2000 he entered the Masters' Program in Computer Science at the University of New Orleans. He worked as a Research Assistant in the Civil and Environmental Engineering Department until May 2001. He also worked as an assistant system administrator in the Math Department for a semester. He then worked as a Teaching Assistant in the Computer Science Department. During his work on his thesis, he developed interest in handheld programming and mobile computing.